LinuxFocus article number 292
http://linuxfocus.org

IDS - Intrusion Detection Systems, Deel I

door Klaus Müller
<Socma(at)gmx.net>

Over de auteur:
Klaus Müller, ook bekend als 'Socma' is nog steeds student en houdt zich bezig met programmeren en beveiliging van Linux.

Vertaald naar het Nederlands door:
Guus Snijders
<ghs(at)linuxfocus.org>

Kort:

Dit is het eerste artikel van serie over Intrusie Detectie Systemen. Het eerste deel zal niet alleen typische IDS architecturen bespreken, maar ook typische aanvallen tegen Intrusie Detectie Systemen.

Introductie

Voor ik begin eerst een korte uitleg, om verkeerd begrip te voorkomen: IDS staat voor Intrusion Detection System (NL: inbraak detectie systeem). In de tekst wordt ook aan IDS gerelateerd als een systeem dat inbraken herkent en begint met tegenmaatregelen (waarbij tegenmaatregelen optioneel zijn). Eerst zullen we de verschillende soorten IDS bekijken om achteraf de verschillende taktieken te bespreken. In het laatste deel zullen we verschillende programma's bekijken, zoals Snort, bofh,... en mijn project COLOID.
Ik zal vast een paar termen uitleggen die niet direct in de tekst worden uitgelegd:

- false positive = een alarm dat een aaval heeft plaatsgevonden, terwijl dit niet het geval was
- false negative = de IDS heeft de aanval niet gedetecteerd en (dus) niet gefilterd

IDS overzicht

Het idee van dit hoofdstuk is om verschillende soorten van Intrusion Detection Systems te bekijken en tegelijkertijd de voor- en nadelen te bespreken. Maar eerst een paar woorden over de betekenis en bedoeling van IDS'en. Een IDS observeert, min of meer, activiteiten op een bepaald network of host.

Met verschillende methoden, die hier besproken worden, probeert het uit te vinden of de beveiliging van een host wordt bedreigd (of er een "aanval" plaatsvindt) om achteraf maatregelen te treffen. Meestal worden er log bestanden aangelegd, en is het een van de belangrijkste doelen om deze te analyseren en te reageren op de indicatie van een inbraak of een illegale poging van een user om meer rechten te verkrijgen.
Simpel gezegd, zijn er de volgende drie gebieden:

- Informatie Bronnen
- Reactie
- Analyse

Reactie en Analyse zullen later exacter worden behandeld, eerst de verschillende soorten "Informatie Bronnen. Vanwege de uitgebreide mogelijkheden om een PC of een netwerk aan te vallen, zijn er verschillende soorten IDS'en (uiteraard kunnen deze gecombineerd worden), welke verschillen in de punten die ze controleren en wat ze controleren (informatie bronnen).

Host - Gebaseerde Intrusie Detectie

Host-gebaseerde Intrusie Detectie Systemen analyseren informatie op een enkele host. Daar ze meestal niet kijken naar het volledige netwerk verkeer, maar "slechts" naar de activiteiten op dezelfde host zijn ze in staat om preciezere verklaringen af te leggen over het soort aanval. In aanvulling daarop kun je direct de impact zien op die bepaalde PC. Bijvoorbeeld dat een bepaalde gebruiker met succes een aanval startte. Host-gebaseerde IDS'en gebruiken meestal twee verschillende bronnen voor om informatie te leveren over activiteiten: systeem logs en besturingssysteem audit trails. Beide hebben voor- en nadelen, daar besturingssysteem audit trails exacter en gedatailleerder zijn en betere informatie over het systeem kunnen leveren, terwijl systeem logs meestal alleen de belangrijke informatie bevatten en daardoor een stuk kleiner zijn. Systeem logs kunnen beter gecontroleerd en geanalyseerd worden vanwege hun omvang.

Voordelen:

- je "ziet" de gevolgen van een aanval en reageert er beter op
- Trojaanse paarden etc. zijn beter te herkennen daar de beschikbare informatie/mogelijkheden uitgebreider zijn
- je kunt aanvallen detecteren die niet voor Netwerk-gebaseerde IDS'en zichtbaar zijn omdat verkeer vaak versleuteld is
- je kunt de activiteiten op je host exact observeren

Nadelen:

- ze zijn niet goed in het herkennen van scans
- ze zijn kwetsbaarder voor DoS aanvallen
- de analyse van besturingssysteem audit trails kost erg veel tijd door de grootte ervan.
- ze vormen een (deels) behoorlijke belasting voor de CPU van de host

Voorbeelden:

- Tripwire [http://www.tripwire.com/products/index.cfml]

- SWATCH [http://freshmeat.net/redir/swatch/10125/url_homepage/swatch]
- DragonSquire [http://www.enterasys.com/ids/squire/]
- Tiger [http://freshmeat.net/redir/tiger-audit/30581/url_homepage/tiger]
- Security Manager [http://www.netiq.com/products/sm/default.asp]

Netwerk Intrusie Detectie (NIDS)

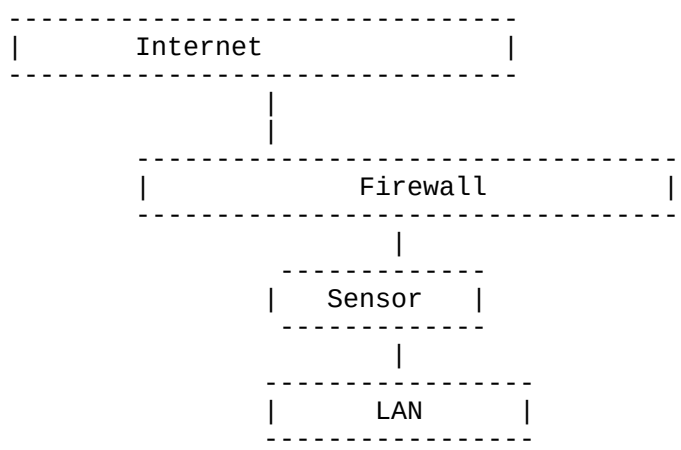
De belangrijkste taak van Netwerk IDS is de analyse en interpretatie van over het netwerk getransporteerde pakketten. Signatures (’handtekeningen’) worden gebruikt om pakketten te beoordelen op "verdachte" inhoud, zoals bijvoorbeeld /etc/passwd. We zullen dit bespreken in de loop van het artikel. Meestal worden er zogenoemde sensors (vaak losse hosts) gebruikt, die niets anders doen dan het analyseren van van verkeer, en, indien nodig, een alarm afgeven. Daat dit hun enige taak is, is het mogelijk om deze sensors beter te beschermen. In deze context komt de zogeheten "Stealth mode" veel voor, dat wil zeggen dat de sensors doen alsof ze "onzichtbaar" zijn, zodat het voor een aanvaller lastiger is ze op te sporen of aan te vallen.

"Stealth mode can be used for network sensors to make the promiscuous mode network interface card (NIC) invisible because it simply doesn’t have an IP address in this mode. This is achieved by using a separate NIC in each network sensor machine to communicate with the console over, usually, a physically isolated secure network. Stealth mode makes it more difficult for a hacker to attack the network sensor itself."

Deze paragraaf (komt uit de beschrijving van RealSecure) legt nogmaals uit wat een sensor in Stealth Mode is. De sensors schakelen in promiscuous mode (eenvoudig gezegt: de mode waarin de netwerk adapter alle netwerk verkeer leest) en heeft geen eigen IP. Hierdoor zou het voor de aanvaller zo lastig mogelijk moeten zijn om de sensor te vinden. Overigens wordt deze mode ook gebruikt door packet sniffers als tcpdump...

In principe kun je sensors onderbrengen in de volgende gebieden:

Binnenin de firewall (versimpeld):

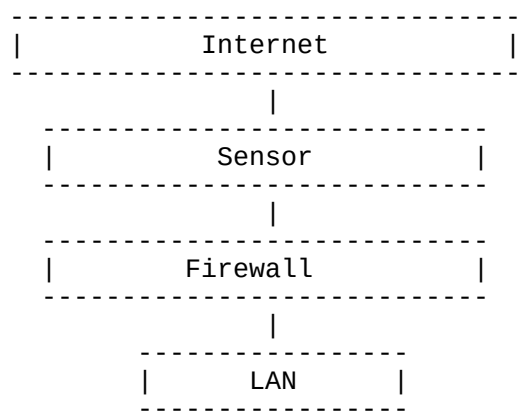


Dit diagram geeft een mogelijke oplossing en geeft alleen aan dat de sensors zich niet tussen DMZ en firewall bevinden. Voor het geval de uitdrukking DMZ je niet bekend is: het staat voor DeMilitarized Zone (gedemilitariseerde zone) en is een gebied dat aan alle kanten beveiligd is, maar wel van buitenaf bereikbaar.

Als de sensors zich in de firewall bevinden, is het eenvoudiger voor ze om te bepalen of een firewall incorrect was geconfigureerd en in aansluiting daarop kom je te weten of een potentiële aanval door de firewall kwam, of niet. Te oordelen naar de ervaring, creëren sensors minder false positives als ze zich "minder afwezig" gedragen en daardoor is het verkeer minder, waarbij de waarschijnlijkheid van een vals alarm afneemt.

Sensors geplaatst in een firewall dienen als instrusie detectie. Als je sensor deze taak vervult, zou je deze binnenin de firewall moeten plaatsen...

Buiten de firewall (versimpeld):



Sensors worden vaak buiten de externe firewall geplaatst zoals weergegeven in het diagram. De reden hiervoor is dat de sensor het volledige verkeer van het Internet kan ontvangen en analyseren. Als je de sensor hier plaatst is het niet zeker dat werkelijk alle aanvallen worden gefilterd en gedetecteerd, zoals bijvoorbeeld TCP aanvallen. In dit geval zou je kunnen proberen de aanval te detecteren met zogenoemde signatures (meer hierover in het gedeelte over signatures). Desalniettemin wordt door veel experts aan deze opzet de voorkeur gegeven omdat er de mogelijkheid is om de aanvallen (op de firewall) te loggen en te analyseren, zo kan de admin zien of hij de firewall configuratie zou moeten wijzigen.

Sensors die buiten de firewall worden geplaatst dienen als aanval detectie (anders dan het plaatsen in de firewall!). Als je sensors deze aanvallen zouden detecteren, zou je ze buiten de firewall moeten plaatsen...

- Buiten en in de firewall

In feite verbindt deze variant beide eerder genoemde varianten. Maar, het gevaar is dat je de sensors en/of de firewall verkeerd configureerd, je kunt niet simpelweg de voordelen van beide varianten aan deze variant toevoegen. Dit zijn niet de enige mogelijkheden om de sensors te plaatsen, je kunt ze, natuurlijk, ergens anders plaatsen, maar de bovengenoemde zijn de meest gebruikte.

Voordelen:

- de sensors kunnen ook beveiligd zijn, terwijl ze "slechts" verkeer monitoren
- je kunt scans beter detecteren - op basis van signatures... je kunt verkeer "filteren" (eigenlijk, zoals we later zullen laten zien, is dit altijd het geval)

Nadelen:

- de waarschijnlijkheid van de zogenoemde false negatives (aanvallen die niet worden herkend als aanvallen) is hoog en het is moeilijk om het hele netwerk te controleren.
- meestal moeten ze opereren op versleutelde pakketten, waar de analyse van pakketten gecompliceerd is
- in tegenstelling tot host-gebaseerde IDS'en zien ze niet de omvang van een aanval

Voorbeelden:

- NetRanger [http://www.cisco.com]
- Dragon [http://www.securitywizards.com]
- NFR [http://www.nfr.net]
- Snort [http://www.snort.org]
- DTK [http://all.net/dtk/dtk.html]
- ISS RealSecure [http://www.uh.edu/infotech/software/unix/realsecure/index.html]

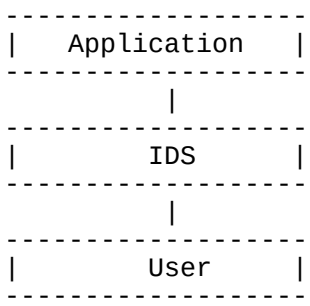
Hoewel ik onderscheid maak tussen HIDS en NIDS, wordt het verschil steeds kleiner, daar ook HIDS de basis functionaliteit van NIDS hebben. Bekende IDS'en zoals die van ISS ReaulSecure noemen zichzelf niet alleen NIDS maar "host-gebaseerd en netwerk-gebaseerd IDS". In de toekomst zal het verschil tussen de twee nog minder duidelijk worden (beide systemen "groeien" steeds meer "naar elkaar toe".

Network Node Intrusion Detection (NNIDS)

In principe werkt dit nieuwe type (NNIDS) als een typisch NIDS, je kunt pakketten uitlezen van netwerk verkeer en deze analyseren. Maar dit betreft alleen de pakketten die geadresseerd zijn aan de netwerk node (dit is waar de naam weg komt). Een ander verschil tussen NNIDS en NIDS is dat NIDS werken in promiscuous mode terwijl NNIDS'en dit niet doen. Daar niet ieder pakket wordt geanalyseerd zal de performance van het systeem er niet zo zwaar onder lijden. Deze systemen werken in de regel erg snel.

Application Based Intrusion Detection

Applicatie gebaseerde IDS'en zijn een subgroep van host-gebaseerde IDS'en maar ik noem ze hier apart. Ze monitoren de interactie tussen gebruiker en programma waarbij vooral extra logbestanden worden aangelegd om informatie te bieden over de activiteiten. Daar je direct tussen gebruiker en programma opereert kun je vrij eenvoudig verdacht gedrag "filteren". Je kunt een ABIDS visualiseren in het volgende diagram:



Voordelen:

- je werkt op een niet-versleuteld niveau, in tegenstelling tot, bijvoorbeeld een Netwerk gebaseerde IDS, waardoor de analyse eenvoudiger wordt
- je kunt verdachte commando’s, die de gebruiker probeert te gebruiken op/met het programma, detecteren en voorkomen

Nadelen:

- geen beveiliging en weinig mogelijkheden om, bijvoorbeeld, Trojaanse paarden te detecteren (je werkt niet in kernel space)
- de log bestanden die door dit soort IDS worden aangelegd zijn doelen voor "aanvallers" en niet zo veilig als, bijvoorbeeld, besturingssysteem audit trails

Stack Based Intrusion Detection

Ook een nieuw soort IDS is de zogenoemde Stack geBaseerd Intrusie Detectie Systeem (SBIDS). Maar op het moment heb ik niet voldoende informatie om over dit type IDS te schrijven. In een toekomstige versie van dit artikel zal de beschrijving zeker volgen...

Honeypot

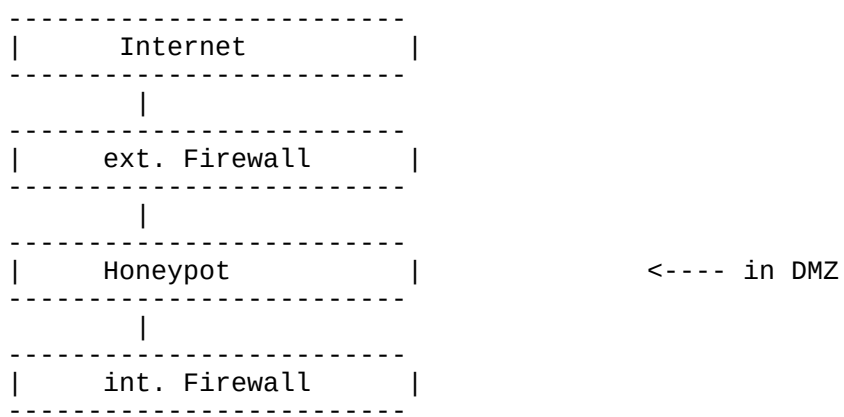
Het woord Honeypot wordt in veel verschillende contexts gebruikt. Om verwarring te voorkomen zal ik bij de beschrijving blijven uit SANS Institute’s Intrusion Deception FAQ:

"Honeypots are programs that simulate one or more network services that you designate on your computer’s ports. An attacker assumes you’re running vulnerable services that can be used to break into the machine. A honeypot can be used to log access attempts to those ports including the attacker’s keystrokes. This could give you advanced warning of a more concerted attack". In some cases, a honeypot is simply a "box".

(NL: Honeypots zijn programma’s die een of meer netwerk services simuleren, die je aangeeft op de poorten van de computer. Een aanvaller neemt aan dat je kwetsbare services draait die gebruikt kunnen worden om in te breken in de machine. Een honeypot kan gebruikt worden om de toegangspogingen tot

deze poorten te loggen, alsmede de toetsaanslagen van de aanvaller. Dit zou je een geavanceerdere waarschuwing op moeten leveren van een gezamelijke aanval. In sommige gevallen is een honeypot slechts een "box".)

Voor de buitenwereld lijkt het kwetsbaar, terwijl het verkeer logt en ook analyseerd. Omdat honeypots kwetsbaar overkomen en verbindingen niet zouden moeten voorkomen, wordt iedere verbinding dus gezien als verdacht.



Het interne netwerk hoeft ook niet te worden blootgesteld omdat een honeypot meestal in de DMZ (DeMilitarized Zone) word geplaatst. De belangrijkste taak van een honeypot bestaat uit het analyseren van verkeer, bijvoorbeeld als bepaalde processen werden gestart, welke bestanden werden veranderd... zodat je een aardig profiel kunt samenstellen van de potentiële aanvallers.

Maar niet alle honeypots zijn hetzelfde. Ze zijn te onderscheiden in drie "varianten" welke de beveiliging van het systeem op verschillende manieren uitbreiden:

Preventie: Detecteren en voorkomen van aanvallen: meldt dat er aanval plaatsvond (mogelijk met een lokalisatie van de aanvaller, wat voor soort aanval, welke services het over ging...?)
Reactie: de (tegen)actie, alleen detectie biedt niks als je er geen stappen tegen onderneemt. Padded Cell Systems bieden speciale mogelijkheden voor tegen-acties. Dus, reactie gaat ervan uit dat je/de IDS reageert op een aanval.

De verschillende reactie opties van IDS zullen we later bekijken... In aanvulling, honeypots kunnen geclassificeerd worden in twee grotere categorieën, onderzoeks- en productie honeypots. Productie honeypots zijn er om potentiële risico’s in een netwerk te verminderen terwijl onderzoeks honeypots dienen om informatie over de aanvallers te verzamelen.

Preventie:
Honeypots zijn zeker niet de meest geschikte systemen om aanvallen af te wenden. Zoals we later zullen zien bij Padded Cell Systems, kan een fout geprogrammeerde en/of fout geconfigureerde honeypot zelfs aanvallen ondersteunen waarbij dit het hele onderwerp van IDS aangaat.

Detectie:
Schijnbaar is dit het grootste voordeel van honeypots daar deze erg goed kunnen zijn in het detecteren van aanvallen. In deze context kunnen ze vooral ook de systeem logs analyseren en intepreteren. Het

plaatsen van een honeypot speelt een beslissende rol, een admin kan alleen profiteren van honeypots als ze juist geconfigureerd en geplaatst zijn. Ze worden vaak geplaatst tussen belangrijke servers om eventuele scans van het hele netwerk te detecteren of in de aanwezigheid van een belangrijke server om illegale toegang met behulp van poort doorsturing te detecteren (bijvoorbeeld, als iemand probeert de server op bepaalde poorten te benaderen, wordt hij doorgestuurd naar de honeypot. Daar deze toegang niet toegestaan zou moeten zijn, zou er een waarschuwing moeten zijn).

Reactie:
In het deel over Antwoorden (lees hieronder), kun je zien welke mogelijkheden een honeypot heeft om te reageren. Bij het gebruik van een honeypot moet je in gedachten houden dat de host zo aantrekkelijk mogelijk voor de aanvaller moet lijken, het zou dus niet te moeilijk moeten zijn om de host aan te vallen. In principe zouden er weinig aanpassingen moeten worden gedaan aan de standaard configuratie, daar teveel aanpassingen alleen maar verdacht lijken. Desalniettemin, moet je niet het originele idee vergeten, dus controleer de toegang, log activiteiten, ... Een benadering die ik verschillende keren heb gehoord is om de honeypot in z’n eigen subnet te plaatsen, achter een firewall. Dit heeft de normale alarm-mogelijkheden en kan waarschuwingen weergeven. Daar de logs gezochte doelen van aanvallers zijn, zou je ze niet op de host zelf moeten opslaan, maar naar een andere server sturen. Soms worden er ook sniffers ingezet om te zoeken naar bepaalde sporen en verkeer te "loggen". Als je genoeg informatie hebt verzameld, zou je de logs moeten analyseren en zoeken naar zwakheden van het netwerk, respectievelijk deze rechtzetten. Vanwege de hoeveelheid informatie zou het eenvoudiger moeten zijn om beveiligingsgaten te dichten en actie tegen aanvallers te ondernemen. Maar je zou wel in de gaten moeten houden dat honeypots niet geheel legaal zijn en dat het kan worden geïntepreteerd kan worden als een "uitnodiging tot aanval" en als je je realiseert dat de host werd aangevallen (en je geen tegenmaatregelen trof), dat ook als een blijk van negeren kan worden gezien. Sommige juridische argumentaties tegen honeypots klinken banaal, maar je zou de wet in jouw plaats moeten controleren. Als iemand een ander netwerk aanvalt, vanuit jouw netwerk en schade aanricht, moet je (waarschijnlijk) betalen wegens de al genoemde redenen.
Verdere onderzoeken wezen uit dat het inzetten van honeypots in Europa geen probleem is (als je een wet tegenkomt die dit tegenspreekt, mail er alsjeblieft over, mijn zoektocht heeft niet zo’n wet opgeleverd...).

Padded Cell Systems

Normaal gesproken werken deze systemen samen met een van de andere systemen. Als de IDS een aanval opmerkt, wordt de aanvaller doorgestuurd naar een "padded cell host". Zodra ze zich in deze padded cell bevinden, kunnen ze geen schade aanrichten, daar de volledige omgeving gesimuleerd is, een "nep" omgeving dus. Het is belangrijk dat deze omgeving zo realistisch mogelijk wordt gepresenteerd, respectievelijk dat de aanvaller denkt dat de aanval geslaagd was (min of meer). Er is een mogelijkheid om iedere actie van de aanvaller te loggen, analyseren en volgen. Het probleem met padded cell systemen is dat ze mogelijk illegaal zijn in bepaalde landen (daar zulke tegen activiteiten kunnen worden gezien als een "aanval", hoewel ze alleen maar bedoeld zijn om informatie te verzamelen over de echte aanvaller).

Voordelen:

- eenmaal in een padded cell host, kunnen aanvallers geen schade aanrichten daar het slechts een "nep" omgeving is
- de admin kan de activiteiten direct volgen/loggen om meer informatie over de aanval en het doel

van aanval te verkrijgen en is dus beter in staat om tegenmaatregelen te treffen

Nadelen:

- mogelijk is het gebruik van padded cell systems niet legaal (hetzelfde als honeypots, in Europa zou dit geen probleem moeten zijn)
- de implementatie van een dergelijk systeem is complex en vereist kennis daar de volledige omgeving correct gesimuleerd moet worden. Als de admin ergens een kleine vergissing maakt, zou dit systeem zeker gaten in de beveiliging kunnen openen.

Typen aanvallen

Alvorens een IDS te ontwikkelen of gebruiken zou je potentiële gevaren moeten specificeren, als ook welke je verwacht. Ondanks de verschillende mogelijkheden voor aanvallen en doelen, kunnen ze gedefiniëerd worden in vier categoriën:

1) Vertrouwdheid
De gevolgen van de aanval zijn dat het gestelde vertrouwen in een bepaalde user veranderen (meestal naar zijn voorkeur ;), bijvoorbeeld dat je je niet meer hoeft te authenticeren voor een bepaald programma... Zulke omstandigheden worden nog steeds misbruikt, vandaag de dag, bijvoorbeeld als er een onderling vertrouwen is tussen twee hosts, als bijvoorbeeld een user op de ene host op de andere kan inloggen zonder wachtwoord (of anders). Als een aanvaller een dergelijk effect kan bereiken met zijn aanval, is het soms erg moeilijk om zo'n "misbruikt" onderling vertrouwen te achterhalen.

2) Integriteit
Als de user belangrijke configuratie en systeem bestanden veranderd, bestaande binaries omwisseld voor zijn eigen... Vaak worden bestaande binaries (zoals /bin/login) omgewisseld door eigen binaries. Hierbij hoeft de user dan alleen maar een vast wachtwoord (in de broncode) op te geven om (meestal) root privileges te krijgen. Dergelijke aanvallen dienen om de eigen privileges te verhogen, bijvoorbeeld om een login achterdeur te installeren. Er bestaan tools als Tripwire, maar niet iedereen gebruikt deze. In aanvulling, er komen vaak verbazingwekkend veel fouten voor in de configuratie en het gebruik, wat Tripwire in zulke gevallen een extra risico maakt.

3) Beschikbaarheid
Hierbij wordt de bereikbaarheid van het systeem aangetast. Dit kan er toe leiden dat bepaalde gebruikers helemaal niet meer kunnen inloggen of dat je alleen kunt inloggen op bepaalde tijden... Het doel is om zoveel mogelijk "ongestoord" te kunnen werken en niet ontdekt kunnen worden door gebruikers.

4) Controle
Als de aanval tot doel heeft om het systeem "over te nemen" dat de controle heeft over bestanden, programma's... Als dit gebeurt, moet je in gedachten houden dat de aanvaller ook alle eerder genoemde mogelijkheden heeft. Als hij totale controle over het systeem heeft, kan hij veranderen, restricties opheffen... wat hij maar wil.

Voor we verder gaan met de verschillende typen aanvallen (DoS, DDos, Scans, ...) zullen we eerst een kleine excursie maken naar de wereld van Integrety Checkers (programma's die de integriteit controleren). Tools als Tripwire vallen onder de host-gebaseerde IDS, het doel van een Integrity Checker is om de integriteit van de verschillende bestanden te controleren en een alarm af te geven als er veranderingen aan een bestand worden geconstateerd.

1) Het aanleggen van een database
De eerste stap na de installatie van een integrity Checker als Tripwire is het aanleggen van een database. Deze stap zou (moet) worden gedaan in een situatie waarin de conditie van het systeem niet is aangetast. Als je de database later aanlegt, (en misschien heeft de aanvaller bestaande binaries veranderd), werkt een Integrity Checker niet zoals hij zou moeten. In zo'n geval worden de veranderde binaries gezien als de "originelen" en als de admin deze omwisseld voor de echte "originelen", wordt er een alarm afgegeven. De meeste tools bieden uitgebreide mogelijkheden om aan te geven van welke bestanden/directories je checksums wilt berekenen. Wij genereren simpelweg een vingerafdruk van het systeem (Tripwire noemt dit Database Initialization Mode).

2) Controle van Integriteit
Als de admin een database heeft (de vingerafdruk) van het systeem, kan hij systeem op ieder gewenst moment controleren. dit wordt gedaan door simpelweg de checksum in de database te vergelijken met de huidige bestanden op de disk. Tripwire biedt meer mogelijkheden. Zie de manpage voor Tripwire of aansluitende documentatie.

Die twee stappen worden gevonden in de meeste Integrity Checkers. Tripwire biedt de mogelijkheid om de database te updaten (als er nieuwe tools zijn geïnstalleerd...) of om het beleidsbestand (policy file), het email notificatie syteem te testen ...

Welke fouten kunnen er gemaakt worden bij het gebruik van Integrity Checkers?

De eerste en grootste fout die een admin kan maken is het creëren van een MD5 hashsum database als bestaande binaries reeds vervangen zijn door een aanvaller. Als een admin aanneemt dat een aanval succesvol was en naderhand een hashsum database aanlegt van zijn "gecompromitteerde" systeem, worden de binaries, vervangen door de aanvaller (bijv. login achterdeur), worden gezien als de "echte" binary. Je zou de database zo snel mogelijk na de installatie moeten aanleggen. Een andere grote fout is wanneer Integrity Checkers (zoals Tripwire) worden gebruikt met de hashsum database op de harde schijf. Op het eerste gezicht lijkt het heel normaal om de database op de harde schijf achter te laten, maar je zou hem alleen daar moeten laten, nadat je hebt gecontroleerd dat de paritie en/of het medium read-only (alleen leesbaar) is. Je moet je ervan verzekeren dat niemand schrijf-toegang kan krijgen tot de database. Als de aanvaller zou kunnen schrijven naar de database, zou hij deze naar wens kunnen aanpassen. Als je al eerder hebt gewerkt met Tripwire, weet je dat in het configuratie bestand kunt opgeven welke bestanden op integriteit moeten worden gecontroleerd. Maar wat als de aanvaller dit configuratie bestand kan lezen/schrijven? Hij zou het zoekpad zo kunnen aanpassen dat de directory met de aangepaste binaries helemaal niet gescand wordt. Het is het beste om het configuratie bestand niet op de harde schijf te laten, en deze op een read-only medium te plaatsen. Sommige mensen zullen denken dat het teveel moeite is om de bestanden op een read-only medium te plaatsen, maar in feite zou je moeten overwegen om meer tijd in beveiliging te steken (een interessant aspect met de gebruikersvriendelijkheid is dat veel beveiligings gaten werden/waren geopend om programma's gebruikersvriendelijker te maken. In ons geval betekend dit dat je minder gebruikers-vriendelijkheid voor een hoger niveau van beveiliging). Tripwire (en andere integrity checkers) zijn zeker capabele

programma’s die de veiligheid kunnen vergroten en potentiële aanvallers kunnen afweren van succesvolle aanvallen. Maar de beste tool is nutteloos als de beveiligings instellingen van het systeem niet in orde zijn. Verwacht dus niet van integrity checkers dat ze al je werk voor je doen.

Een nieuwer type integrity checkers zijn de zogenoemde Realtime Integrity Checkers. In contrast tot "normale" checkers, controleren deze de inegreriteit van de bestanden in realtime. Hier is een kort schema:

1) Ook hier is de eerste stap het aanleggen van een hashsum database

2) Voordat iemand een programma kan uitvoeren wordt de hashsum van het bestand vastgesteld. Als de waarde niet overeenkomt met degene in de database, was de binary vervangen. Als dit het geval is, wordt de uitvoering gestopt.
Dit concept is alleen mogelijk voor realtime integrity checkers (of in ieder geval stel ik me dit concept zo voor). Als een verschil met integrity checkers, verlaat je je niet op de admin om regelmatig de bestanden te controleren, je probeert de correctheid van de binary voor uitvoeren vast te stellen en, indien nodig, de toegang tot deze te blokkeren...

Behalve de bovengoemde categoriëen (ingegriteit, controle, ...) kunnen aanvallen ook anders onderscheiden worden, zoals hoe ze aanvallen, en wat het betekend. Er zijn er natuurlijk vele mogelijkheden, maar de meest voorkomende aanvallen tegen IDS’en zijn:

- Scans
Vandaag de dag vormen scans de gewone "aanvallen", het probleem hier is dat de IDS niet te veel false positives moet genereren. Meestal dienen scans om (meer) informatie over een host/netwerk te krijgen (bijvoorbeeld om bepaalde aanvallen te lanceren). Welke informatie de aanvaller kan krijgen over je eigen netwerk kun je zien met het volgende scan resultaat (nmap - slechts een kort voorbeeld, geeft niet alle mogelijkheden weer):

```
....
Host (192.168.0.0) seems to be a subnet broadcast address (returned 1
extra pings).
Skipping host. Interesting ports on playground.yuma.net 192.168.0.1):
Port           State           Protocol                Service
22             open            tcp             ssh
111            open            tcp             sunrpc
635            open            tcp             unknown
1024           open            tcp             unknown
2049           open            tcp             nfs

TCP Sequence Prediction: Class = random positive increments
                         Difficulty=3916950 (Good luck!)
Remote operating system guess:
       Linux 2.1.122 - 2.1.132; 2.2.0-pre1 - 2.2.2

Interesting ports on vectra.yuma.net (192.168.0.5):
Port           State           Protocol                Service
13             open            tcp             daytime
21             open            tcp             ftp
22             open            tcp             ssh
23             open            tcp             telnet
37             open            tcp             time
79             open            tcp             finger
```

```
111             open            tcp             sunrpc
113             open            tcp             auth
513             open            tcp             login
514             open            tcp             shell

TCP Sequence Prediction: Class = random positive increments
                         Difficulty = 17719 (Worthy challenge)
Remote operating system guess: OpenBSD 2.2. - 2.3

Nmap run completed -- 256 IP addresses (2 hosts up) scanned in 6 seconds
```

Meestal kun je het volgende achterhalen:

- welk besturingssysteem?
- welke versies van bepaalde programma's zijn actief?
- welke services, die je eventueel kunt "aanvallen" draaien er?
- welke poorten staan open
- ...

Door veel sterk verschillende scan technieken samen te gebruiken, is het mogelijk veel informatie te verkrijgen, die de aanvallen kan gebruiken om z'n aanval in te zetten. Ik zal hier echter niet alle technieken bespreken/noemen, maar een paar van de meestgebruikte (voor vragen over bepaalde protocollen... zie de respectievelijke RFCs):

Ping scanning:
Ping scans worden gebruikt om te achterhalen welke hosts online zijn. Hiervoor stuur je de host (of hosts) een ICMP diagram type 8 (echo request) en wacht op een ICMP antwoord diagram type 0 (echo reply). Soms stuur je niet alleen een echo request maar aansluitend een ACK daar ICMP soms geblokkeerd wordt. Als er een RST antwoord komt, is de host online.

Nmap parameter: -sP

TCP Scanning (Vanilla) :
Met TCP scanning probeer je (meestal) te connect()'en naar alle poorten met de drie-voudige begroeting, in andere woorden, je zette een verbinding op naar de poorten (de connecties naar de poorten hadden succes) de antwoord waarde van connect() wordt gecontroleerd. Voor de aanvaller betekend de antwoord waarde of de gebruikte poort (of poorten) open of gesloten zijn.

Nmap parameter: -sT

De volgende output van nmap is van een mijn hosts, direct na een default installatie. Ik heb met opzet geen wijzigingen aangebracht (in de configuratie):

```
 [Socma]$ nmap -sT localhost

Starting nmap V. 2.54BETA36 ( www.insecure.org/nmap/ )
Interesting ports on Diablo (127.0.0.1):
(The 1552 ports scanned but not shown below are in state: closed)
Port       State       Service
21/tcp     open        ftp
23/tcp     open        telnet
80/tcp     open        http
111/tcp    open        sunrpc
```

```
113/tcp    open        auth
6000/tcp   open        X11

Nmap run completed -- 1 IP address (1 host up) scanned in 1 second
```

Een deel van een tcpdump trace (van deze scan:

```
 02:10:15.804704 Diablo > Diablo: icmp: echo request
                          4500 001c 2db8 0000 3501 5a27 7f00 0001
                          7f00 0001 0800 fc95 fb69 0000
 02:10:15.814704 Diablo > Diablo: icmp: echo reply (DF)
                          4500 001c 0000 4000 ff01 7dde 7f00 0001
                          7f00 0001 0000 0496 fb69 0000
 02:10:15.814704 Diablo.58725 > Diablo.http: . ack 110306597 win 3072
                          4500 0028 d223 0000 2a06 c0aa 7f00 0001
                          7f00 0001 e565 0050 ad48 0003 0693 2525
                          5010 0c00 e718 0000
 02:10:15.814704 Diablo.http > Diablo.58725: R 110306597:110306597(0)
         win 0 (DF)
                          4500 0028 0000 4000 ff06 7dcd 7f00 0001
                          7f00 0001 0050 e565 0693 2525 0000 0000
                          5004 0000 a070 0000
 02:10:16.114704 Diablo.1727 > Diablo.821: S 196002918:196002918(0)
 win 32767 <mss 16396,sackOK,timestamp 213509[|tcp]> (DF)
                          4500 003c 8663 4000 4006 b656 7f00 0001
                          7f00 0001 06bf 0335 0bae c466 0000 0000
                          a002 7fff 739c 0000 0204 400c 0402 080a
                          0003 4205
 02:10:16.114704 Diablo.821 > Diablo.1727: R 0:0(0) ack 196002919
        win 0 (DF)
                          4500 0028 0000 4000 ff06 7dcd 7f00 0001
                          7f00 0001 0335 06bf 0000 0000 0bae c467
                          5014 0000 d7c4 0000
 02:10:16.114704 Diablo.1728 > Diablo.440: S 195504823:195504823(0)
 win 32767 <mss 16396,sackOK,timestamp 213509[|tcp]> (DF)
                          4500 003c 68b2 4000 4006 d407 7f00 0001
                          7f00 0001 06c0 01b8 0ba7 2ab7 0000 0000
                          a002 7fff 0ecf 0000 0204 400c 0402 080a
                          0003 4205
```

UDP scanning:
De bedoeling van UDP scans zijn analoog aan TCP scans, daar je wilt achterhalen welke UDP poorten open zijn. Een scan verloopt anders omdat UDP een connectieloos (connectionless) protocol is (TCP is connectie-georiënteerd). UDP scanning "gebruikt" ICMP, als je naar een 0 byte UDP pakket naar de poort stuurt om te wachten op een ICMP "antwoord". Als er een notificatie komt dat de poort onbereikbaar is ("Port unreachable"/code waarde 3), betekend dit dat de poort gesloten is. Als de respectievelijke admin bepaalde services uitschakeld, bijvoorbeeld in /etc/inetd.conf, en je probeert een pakket naar de gebruikte poort te sturen, resulteerd dit in het fout bericht "Port Unreachable"...

Nmap parameter: -sU

```
 [Socma]$ nmap -sU localhost

Starting nmap V. 2.54BETA36 ( www.insecure.org/nmap/ )
Interesting ports on Diablo (127.0.0.1):
```

```
(The 1459 ports scanned but not shown below are in state: closed)
Port       State       Service
111/udp    open        sunrpc

Nmap run completed -- 1 IP address (1 host up) scanned in 4 seconds
```

De bijbehorende tcpdump trace:

```
10:41:55.954397 Diablo > Diablo: icmp: echo request
                         4500 001c cc8f 0000 2801 c84f 7f00 0001
                         7f00 0001 0800 8471 738e 0000
10:41:55.954397 Diablo > Diablo: icmp: echo reply (DF)
                         4500 001c 0000 4000 ff01 7dde 7f00 0001
                         7f00 0001 0000 8c71 738e 0000
10:41:55.964397 Diablo.63793 > Diablo.http: . ack 994287972 win 2048
                         4500 0028 79e3 0000 2506 1deb 7f00 0001
                         7f00 0001 f931 0050 06d8 0003 3b43 a164
                         5010 0800 cccd 0000
10:41:55.964397 Diablo.http > Diablo.63793: R 994287972:994287972(0)
     win 0 (DF)
                         4500 0028 0000 4000 ff06 7dcd 7f00 0001
                         7f00 0001 0050 f931 3b43 a164 0000 0000
                         5004 0000 dbb4 0000
10:41:56.274397 Diablo.63773 > Diablo.15: udp 0
                         4500 001c 8a0b 0000 3011 02c4 7f00 0001
                         7f00 0001 f91d 000f 0008 08af
10:41:56.274397 Diablo > Diablo: icmp: Diablo udp port 15
     unreachable (DF) [tos 0xc0]
                         45c0 0038 0000 4000 ff01 7d02 7f00 0001
                         7f00 0001 0303 fb18 0000 0000 4500 001c
                         8a0b 0000 3011 02c4 7f00 0001 7f00 0001
                         f91d 000f
10:41:56.274397 Diablo.63773 > Diablo.1459: udp 0
                         4500 001c 6c2f 0000 3011 20a0 7f00 0001
                         7f00 0001 f91d 05b3 0008 030b
10:41:56.274397 Diablo > Diablo: icmp: Diablo udp port 1459
     unreachable (DF) [tos 0xc0]
                         45c0 0038 0100 4000 ff01 7c02 7f00 0001
                         7f00 0001 0303 fb18 0000 0000 4500 001c
                         6c2f 0000 3011 20a0 7f00 0001 7f00 0001
                         f91d 05b3
```

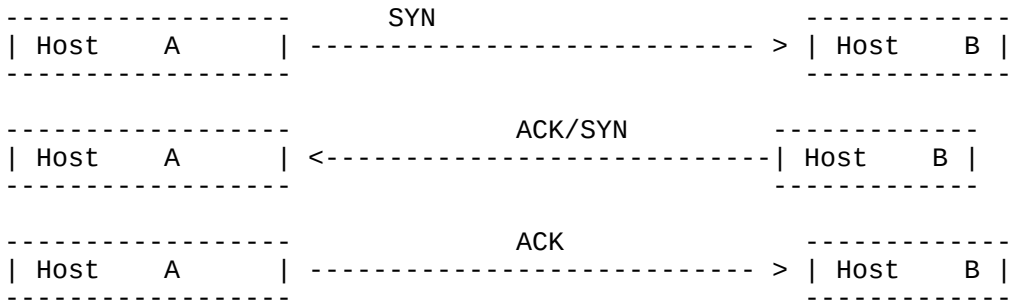
Een andere variant van een UDP scan (UDPrecvfrom() en write() scan) bestaat uit het twee maal scannen van iedere poort. Deze methode gebruikt ICMP met "Port Unreachable", maar alleen root ontvangt dit bericht. Als je een gesloten poort twee keer scant, krijg je, na de tweede scan, "Error 13 : Try Again"...

ACK scanning:
Met het sturen van een ACK pakket naar een poort van een firewall kun je achterhalen welke poorten worden gefilterd en welke niet. Als je een RST antwoord krijgt, betekend het dat de geprobeerde poort "onbeschermd" is, respectievelijk dat deze niet gefilterd wordt, anders krijg je een ICMP fout melding. Dus, je achterhaald niet welke poorten open zijn, maar je verkrijgt preciezere informatie over de firewall (en zijn configuratie).

Nmap parameter : -sA

```
 [Socma]$ nmap -sA localhost

Starting nmap V. 2.54BETA36 ( www.insecure.org/nmap/ )
All 1558 scanned ports on Diablo (127.0.0.1) are: UNfiltered

Nmap run completed -- 1 IP address (1 host up) scanned in 6 seconds

 The tcpdump trace:

10:45:51.864397 Diablo > Diablo: icmp: echo request
                         4500 001c 1617 0000 3901 6dc8 7f00 0001
                         7f00 0001 0800 113d e6c2 0000
10:45:51.864397 Diablo > Diablo: icmp: echo reply (DF)
                         4500 001c 0000 4000 ff01 7dde 7f00 0001
                         7f00 0001 0000 193d e6c2 0000
10:45:51.864397 Diablo.53119 > Diablo.http: . ack 2682022466 win 3072
                         4500 0028 0dda 0000 3206 7cf4 7f00 0001
                         7f00 0001 cf7f 0050 0650 0003 9fdc 6a42
                         5010 0c00 c590 0000
10:45:51.864397 Diablo.http > Diablo.53119: R 2682022466:2682022466(0)
     win 0 (DF)
                         4500 0028 0000 4000 ff06 7dcd 7f00 0001
                         7f00 0001 0050 cf7f 9fdc 6a42 0000 0000
                         5004 0000 d7ef 0000
10:45:52.164397 Diablo.53099 > Diablo.14: . ack 2457451592 win 3072
                         4500 0028 218d 0000 3206 6941 7f00 0001
                         7f00 0001 cf6b 000e 5938 4710 9279 bc48
                         5010 0c00 e74d 0000
10:45:52.164397 Diablo.14 > Diablo.53099: R 2457451592:2457451592(0)
     win 0 (DF)
                         4500 0028 0000 4000 ff06 7dcd 7f00 0001
                         7f00 0001 000e cf6b 9279 bc48 0000 0000
                         5004 0000 93a2 0000
10:45:52.164397 Diablo.53099 > Diablo.imap3: . ack 2457451592 win 3072
                         4500 0028 a075 0000 3206 ea58 7f00 0001
                         7f00 0001 cf6b 00dc 5938 4710 9279 bc48
                         5010 0c00 e67f 0000
```

Stealth scanning (NULL, XMAS, FIN, SYN, ...): Met stealth scanning "misbruik" je de drie-voudige-begroeting. Ik presenteer de drie-voudige-begroeting kort:



Het probleem van TCP scans is dat ze erg verdacht zijn (daar er iedere keer een drie-voudige-begroeting plaastvindt). Met stealth scanning gebeurt er echter het volgende:

Dit diagram lijkt op de three-way-handshake maar met een basis verschil: het weergegeven diagram heeft geen verbinding tussen A en B, Host B zou dus denken dat de verbinding bestaat, alleen bestaat de verbinding niet tot Host A een verdere ACK stuurt naar B (dit wordt ook wel een "half-open" poort genoemd...). De hierboven weergegeven SYN scan impliceert dat de poort van de doel host open is (vanwege de ACK/SYN), als deze gesloten zou zijn, zou je een RST/ACK terug krijgen.

Nmap parameter : -sS

```
 [Socma]$ nmap -sS localhost

Starting nmap V. 2.54BETA36 ( www.insecure.org/nmap/ )
Interesting ports on Diablo (127.0.0.1):
(The 1552 ports scanned but not shown below are in state: closed)
Port       State       Service
21/tcp     open        ftp
23/tcp     open        telnet
80/tcp     open        http
111/tcp    open        sunrpc
113/tcp    open        auth
6000/tcp   open        X11

Nmap run completed -- 1 IP address (1 host up) scanned in 3 seconds

Tcpdump trace:

10:47:41.674397 Diablo > Diablo: icmp: echo request
                          4500 001c 8f08 0000 3501 f8d6 7f00 0001
                          7f00 0001 0800 99a9 5e56 0000
10:47:41.674397 Diablo > Diablo: icmp: echo reply (DF)
                          4500 001c 0000 4000 ff01 7dde 7f00 0001
                          7f00 0001 0000 a1a9 5e56 0000
10:47:41.674397 Diablo.58038 > Diablo.http: . ack 1561498752 win 3072
                          4500 0028 afe5 0000 3206 dae8 7f00 0001
                          7f00 0001 e2b6 0050 82b0 0003 5d12 9480
                          5010 0c00 4e85 0000
10:47:41.674397 Diablo.http > Diablo.58038: R 1561498752:1561498752(0)
     win 0 (DF)
                          4500 0028 0000 4000 ff06 7dcd 7f00 0001
                          7f00 0001 0050 e2b6 5d12 9480 0000 0000
                          5004 0000 dd44 0000
10:47:41.984397 Diablo.58018 > Diablo.1488: S 2803535203:2803535203(0)
     win 3072
                          4500 0028 a4f5 0000 3206 e5d8 7f00 0001
                          7f00 0001 e2a2 05d0 a71a 8d63 0000 0000
                          5002 0c00 88ef 0000
10:47:41.984397 Diablo.1488 > Diablo.58018: R 0:0(0) ack 2803535204
     win 0 (DF)
                          4500 0028 0000 4000 ff06 7dcd 7f00 0001
                          7f00 0001 05d0 e2a2 0000 0000 a71a 8d64
                          5014 0000 94dc 0000
```

Nu, andere scans doen ook mee: FIN scanning, NULL scanning en XMAS scanning. FIN scanning

stuurt alleen een FIN bericht naar de "doel host", al bestaat er geen verbinding tussen hun. Op een gesloten poort wordt een RST terug gestuurd, verder gebeurt er niets.

Nmap parameter : -sF

```
 [Socma]$ nmap -sF localhost

Starting nmap V. 2.54BETA36 ( www.insecure.org/nmap/ )
Interesting ports on Diablo (127.0.0.1):
(The 1552 ports scanned but not shown below are in state: closed)
Port       State       Service
21/tcp     open        ftp
23/tcp     open        telnet
80/tcp     open        http
111/tcp    open        sunrpc
113/tcp    open        auth
6000/tcp   open        X11

Nmap run completed -- 1 IP address (1 host up) scanned in 6 seconds

Tcpdump trace:

10:48:28.704397 Diablo > Diablo: icmp: echo request
                          4500 001c b29d 0000 3401 d641 7f00 0001
                          7f00 0001 0800 a1a7 5658 0000
10:48:28.704397 Diablo > Diablo: icmp: echo reply (DF)
                          4500 001c 0000 4000 ff01 7dde 7f00 0001
                          7f00 0001 0000 a9a7 5658 0000
10:48:28.704397 Diablo.52201 > Diablo.http: . ack 2873378189 win 4096
                          4500 0028 cbeb 0000 2b06 c5e2 7f00 0001
                          7f00 0001 cbe9 0050 9020 0003 ab44 458d
                          5010 1000 54a3 0000
10:48:28.704397 Diablo.http > Diablo.52201: R 2873378189:2873378189(0)
     win 0 (DF)
                          4500 0028 0000 4000 ff06 7dcd 7f00 0001
                          7f00 0001 0050 cbe9 ab44 458d 0000 0000
                          5004 0000 f4d2 0000
10:48:29.004397 Diablo.52181 > Diablo.233: F 0:0(0) win 4096
                          4500 0028 10c6 0000 2b06 8108 7f00 0001
                          7f00 0001 cbd5 00e9 0000 0000 0000 0000
                          5001 1000 d522 0000
10:48:29.004397 Diablo.233 > Diablo.52181: R 0:0(0) ack 1 win 0 (DF)
                          4500 0028 0000 4000 ff06 7dcd 7f00 0001
                          7f00 0001 00e9 cbd5 0000 0000 0000 0001
                          5014 0000 e50e 0000
```

NULL en XMAS scans zijn vooral interessant (vooral met een practische implementatie van protocol anomalie detectie). De naam XMAS (kerstboom) komt van het feit dat alle vlaggen zijn gezet: SYN, ACK, FIN, URG, PUSH. Net als met FIN scanning komt er een RST terug als de poort gesloten is.

Nmap parameter : -sX

```
 [Socma]$ nmap -sX localhost

Starting nmap V. 2.54BETA36 ( www.insecure.org/nmap/ )
Interesting ports on Diablo (127.0.0.1):
(The 1552 ports scanned but not shown below are in state: closed)
Port       State       Service
```

```
21/tcp     open        ftp
23/tcp     open        telnet
80/tcp     open        http
111/tcp    open        sunrpc
113/tcp    open        auth
6000/tcp   open        X11

Nmap run completed -- 1 IP address (1 host up) scanned in 5 seconds

Tcpdump trace:

10:44:24.004397 Diablo > Diablo: icmp: echo request
                         4500 001c ffcc 0000 2a01 9312 7f00 0001
                         7f00 0001 0800 103d e7c2 0000
10:44:24.004397 Diablo > Diablo: icmp: echo reply (DF)
                         4500 001c 0000 4000 ff01 7dde 7f00 0001
                         7f00 0001 0000 183d e7c2 0000
10:44:24.004397 Diablo.36398 > Diablo.http: . ack 718216305 win 2048
                         4500 0028 2e28 0000 2906 65a6 7f00 0001
                         7f00 0001 8e2e 0050 9220 0003 2acf 1c71
                         5010 0800 41f0 0000
10:44:24.004397 Diablo.http > Diablo.36398: R 718216305:718216305(0)
     win 0 (DF)
                         4500 0028 0000 4000 ff06 7dcd 7f00 0001
                         7f00 0001 0050 8e2e 2acf 1c71 0000 0000
                         5004 0000 dc1f 0000
10:44:24.304397 Diablo.36378 > Diablo.1996: FP 0:0(0) win 2048 urg 0
                         4500 0028 7651 0000 2906 1d7d 7f00 0001
                         7f00 0001 8e1a 07cc 0000 0000 0000 0000
                         5029 0800 13d3 0000
10:44:24.304397 Diablo.1996 > Diablo.36378: R 0:0(0) ack 1 win 0 (DF)
                         4500 0028 0000 4000 ff06 7dcd 7f00 0001
                         7f00 0001 07cc 8e1a 0000 0000 0000 0001
                         5014 0000 1be7 0000
```

De andere mogelijkheid, genaamd NULL scan, betekend dat er geen vlag gezet is, als de poort gesloten is, wordt er geantwoord met een RST.

Nmap parameter : -sN

```
 [Socma]$ nmap -sN localhost

Starting nmap V. 2.54BETA36 ( www.insecure.org/nmap/ )
Interesting ports on Diablo (127.0.0.1):
(The 1552 ports scanned but not shown below are in state: closed)
Port       State       Service
21/tcp     open        ftp
23/tcp     open        telnet
80/tcp     open        http
111/tcp    open        sunrpc
113/tcp    open        auth
6000/tcp   open        X11

Nmap run completed -- 1 IP address (1 host up) scanned in 5 seconds

Tcpdump trace:

10:43:37.594397 Diablo > Diablo: icmp: echo request
                         4500 001c 2ecf 0000 2c01 6210 7f00 0001
                         7f00 0001 0800 8f87 6878 0000
```

```
10:43:37.594397 Diablo > Diablo: icmp: echo reply (DF)
                         4500 001c 0000 4000 ff01 7dde 7f00 0001
                         7f00 0001 0000 9787 6878 0000
10:43:37.604397 Diablo.34607 > Diablo.http: . ack 1932747046 win 4096
                         4500 0028 ee0f 0000 3706 97be 7f00 0001
                         7f00 0001 872f 0050 5b20 0003 7333 6126
                         5010 1000 ead5 0000
10:43:37.604397 Diablo.http > Diablo.34607: R 1932747046:1932747046(0)
     win 0 (DF)
                         4500 0028 0000 4000 ff06 7dcd 7f00 0001
                         7f00 0001 0050 872f 7333 6126 0000 0000
                         5004 0000 5605 0000
10:43:37.904397 Diablo.34587 > Diablo.408: . win 4096
                         4500 0028 e3bb 0000 3706 a212 7f00 0001
                         7f00 0001 871b 0198 0000 0000 0000 0000
                         5000 1000 192f 0000
10:43:37.904397 Diablo.408 > Diablo.34587: R 0:0(0) ack 0 win 0 (DF)
                         4500 0028 0000 4000 ff06 7dcd 7f00 0001
                         7f00 0001 0198 871b 0000 0000 0000 0000
                         5014 0000 291b 0000
```

Je hoeft de three-way-handshake niet te voltooien, dus stealth scanning (zoals genoemd) is minder verdacht dan TCP scanning. IDS'en zouden deze afwijkingen in alle gevallen (XMAS en NULL) moeten detecteren...

FTP bounce:
Met sommige ftpds kan het PORT commando misbruikt worden om een willekeurige verbinding op te zetten vanaf de ftp server naar een andere machine. Maar eerst een kort overzicht hoe dit "normaal" gebeurt. Eerst maakt de client een verbinding naar de ftp server (poort 21), de ftp server "creëert" een tweede verbinding terug naar de client (om in staat te zijn data terug te sturen). Voor deze tweede verbinding gebruik je het PORT commando. Het interessante hier is dat het commando het IP nummer en de poort (welke gebruikt dient te worden) van de client aangeeft. Hierop creëert de server een verbinding vanaf bronpoort 20, de doelpoort is degene die werd gespecificeerd door de het PORT commando. Het aanvalsdoel is het PORT commando, waarmee je kunt manipuleren naar welke poort van de (vermoede) client moet worden verbonden, bijvoorbeeld naar het slachtoffer in plaats van onze host. Nadat het IP en de poort aangepast zijn kun je het eigenlijke data verkeer initiëren met een "list" of "get". Nu, je kunt het antwoord van ftp controleren, als we een "425: Can't build data connection: Connection refused" krijgen, is deze poort gesloten. Als we een "150 : File status okay about to open data connection" of "226: Closing data connection. Requested file action successful (bijvoorbeeld een file transfer of file abort)" inplaats als antwoord, weten we dat de gespecificeerde poort open is.

Nmap parameter: -b

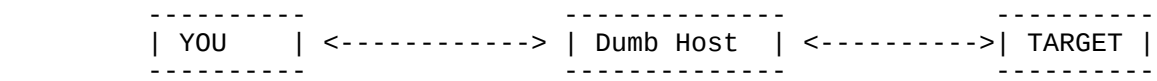
Fragmented packets: Deze methode gebruikt de fragmentatie van een IP pakket bij TCP. Normaal gesproken treedt fragmentatie op als de datagrammen groter zijn dan de maximaal toegestane grootte, deze grootte limitatie is genaamd MTU (Maximum Transmission Unit). Gefragmenteerde pakketten worden weer bij elkaar geplaatst aan het einde van een node. Dit gedrag kan misbruikt worden. Niet iedere IDS/Firewall werkt met fragmentatie, er treden soms fouten op met gefragmenteerde pakketten. In plaats van ons pakket normaal te versturen, delen we deze op (in fragmenten). Deze bevatten "gewone" data als bron IP, doel IP, bron poort... Het is mogelijk dat de gebruikte firewall/IDS problemen heeft met het assembleren van deze fragmenten. Deze problemen kunnen op verschillende manieren optreden, het kan voor een crash van het hele systeem zorgen, of het pakket wordt doorgelaten. Ons pakket kan mogelijk doorgelaten worden omdat het assembleren fouten opleverde en

het pakket foutief werd aangemerkt als "onschuldig". Soms worden niet alle fragmenten goed gecontroleerd, er wordt bijvoorbeeld alleen een bepaald pakket gecontroleerd, en de rest wordt doorgelaten. Met deze techniek kun je minder opvallend scannen daar het verkeer wordt gemarkeerd als "onschuldig" en dus geen alarm oplevert. Aan de andere kant is deze theory afhankelijk van de problemen die de IDS (firewall) mogelijk heeft met het verwerken en samenvoegen van fragmenten.

Reverse Ident Scanning:
Eerst een citaat uit RFC 1413, de RFC voor het Identificatie Protocol:

```
"The Identification Protocol (a.k.a.,
"ident", a.k.a., "the Ident Protocol") provides a means to
determine the identity of a user of a particular TCP
connection. Given a TCP port number pair, it returns a
character string which identifies the owner of that connection
on the server's system."



(NL: het Identification Protocol (ook bekend als "ident") biedt een
mogelijkheid de identiteit van een gebruiker op een bepaalde TCP
connectie te achterhalen. Met behulp van een TCP poort nummer paar,
retourneerd het een karakter string die de eigenaar van die connectie
naar het server systeem weergeeft.)
```

Reverse Ident Scanning gebruikt identd om te vragen naar de eigenaar van een draaiend proces. Deze techniek wordt vooral gebruikt om daemons te vinden die draaien als root, om vervolgens precies deze daemons aan te vallen.



Hier zou Dump Host ("Domme Host") zo min mogelijk verkeer moeten hebben. De reden hiervoor zal later duidelijk worden. Waarom wordt Dump Host gebruikt, waarom hebben we er een nodig? Ok, deze vraag leidt tot de eigenlijke aanval en met dit ook de uitleg waarom een dump host nodig is. Om uit te vinden of een poort van "TARGET" open of gesloten is, moet je het IP ID veld van Dump Host bekijken. Hiervoor wordt een pakket (echo request) gestuurd naar Dump Host, en met het antwoord hierop kun je het ID veld lezen, resp. de waarde van het ID veld. Vervolgens stuur je TARGET een pakket waar het bron adres die van Dump Host is. Het antwoord van TARGET wordt dan ontvangen door onze Dump Host. Als hij een SYN/ACK ontvangt van TARGET, betekend dat dat de poort open is. Als dump host een RST/ACK krijgt van TARGET, sturen we Dump Host een andere ping. Als de poort van target open was en een RST terug stuurde, zal het IP ID veld van Dump Host worden verhoogd, met de poort gesloten gebeurt er niets. Door de nieuwe IP ID waarde van Dump Host uit te lezen, kunnen we achterhalen of een poort van TARGET open of gesloten was. Hopelijk is het nu duidelijk waarom we een dump host nodig hebben, dus een host met weinig verkeer. Als er teveel verkeer plaatsvindt op de host, wordt het lastiger om aan te geven welke poorten open staan (op TARGET), de waarschijnlijkheid van het fout hebben is veel hoger bij meer verkeer.

Fingerprint OS detection:
Fingerprint OS detection dient om het besturingssysteem van de server te achterhalen. De meeste nieuwe scanners leveren niet alleen bijv. "Linux" of "Solaris" op als antwoord, maar een specificatie van

versies (van het besturingssysteem). Hiervoor kun je een "vingerafdruk" (profiel) maken van het besturingssysteem. Tegenwoordig kun je de banners van telnet, ftp (zie boven) niet meer vertrouwen, daar deze veranderd kunnen worden. Als de aanvaller kan achterhalen welke exacte versie de doelhost gebruikt, kan hij scipts, exploits, ed bouwen, om verder te gaan met zijn aanval. Deze techniek maakt gebruik van het feit dat besturingssystemen in veel details van elkaar verschillen (wat een inzicht ;). Daar er veel documenten bestaan over deze techniek, zal ik een kort overzicht geven van de belangrijkste test mogelijkheden:

- FIN test: Als een host een FIN pakket ontvangt op een open poort zou hij geen antwoord moeten geven (RFC 793), maar er bestaan uitzonderingen, bijvoorbeeld MS Windows, BSDI, CISCO, MPS, ... welke een RST sturen als antwoord.

-ACK sequential numbers: Als een FIN | PSH | URG naar een gesloten poort wordt gestuurd, wordt het sequentie nummer van het ACK pakket meestal gezet op het eigen sequentie nummer. Maar er zijn ook Windows versies in omloop die een uitzondering maken ;). Windows stuurt het eigen sequentie nummer +1 terug ...

-BOGUS flag test: Als er een niet-gedefiniëerde TCP flag wordt gebruikt in de TCP header (in een SYN pakket), Linux hosts < 2.0.35 'adopteren' deze flag instellingen. Andere OS'en resetten bij het ontvangen van een SYN+BOGUS pakket...

-TCP initial window: De meeste besturingssystemen gebruiken (meestal) constante window groottes (van de reply pakketten). AIX gebruikt bijvoorbeeld 0x3F25, MS NT5, OpenBSD and FreeBSD use 0x402E....

-Don't fragment bit test: Sommige besturingssystemen verschillen in het zetten van deze bit in sommige pakketten of niet. Dus kun je aansluitend onderscheiden welk OS beschikbaar is...

-TCP options test: De basis van deze test is snel verteld: je stuurt een vraag naar de host, zet diverse opties en controleert het antwoord of dezelfde opties zijn gezet. Die opties die in het antwoord voorkomen zijn ondersteund... Zoals het geval kan zijn met het besturingssysteem (en versie) in kwestie, worden bepaalde opties in principe niet ondersteund, andere wel. Dus het gebruikte besturingssysteem kan nauwkeuriger worden aangegeven.

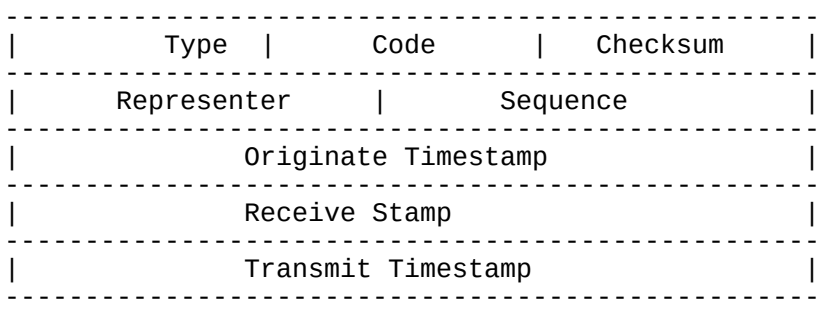
Nmap parameter : -O

Er zijn veel testen hier niet besproken zijn, maar op het net zijn er genoeg documenten te vinden over fingerprint OS detection. Fyodor (NMAP) heeft een korte paper hierover geschreven. Dit deel geeft alleen een kort overzicht van veel gebruikte scan technieken. Voor iemand die werkt met protocol anomalie detectie zijn er zeker enkele begin punten (bepaalde flag combinaties welke je kunt beschouwen als abnormaal...).

Ander ICMP gerelateerd spul
Behalve de genoemde echo reply/request, ondersteund ICMP verdere bericht typen waarmee je verder informatie over het netwerk kunt verzamelen (zogenoemde NON - ECHO - REQUESTS):

ICMP Time Stamp Request / Reply (RFC 792):
De eigenlijke betekenis van Time Stamp Requests (type 13) is om de tijds instellingen van een remote

system te achterhalen. Als de remote host een Time Stamp Request krijgt, stuurt het een Time Stamp Reply (type 14) terug. Eerst de structuur van een time stamp (relatief naar RFC):



Voordat ik begin met het gebruik van een Time Stamp Request voor een aanvaller, vertel ik je de basics van de time stamp. Voor ons zijn alleen de laatste drie "velden" van belang. Hier de sectie uit de RFC:

```
The Originate Timestamp is the time the sender last touched the
message before sending it, the Receive Timestamp is the time the
echoer first touched it on receipt, and the Transmit Timestamp is
the time the echoer last touched the message on sending it."

(NL: De Originate Timestamp is de laatste keer dat de zender het
bericht 'raakte' voor deze te versturen, de Receive Timestamp is
de tijd dat echoer het voor het eerst 'raakte' bij ontvangen, en
de Transmit Timestamp is de tijd dat de echoer het bericht voor
het laatst 'raakte' alvorens deze te versturen.)
```

Zoals verteld in de RFC is de teruggestuurde timestamp het aantal miliseconden sinds middernacht UT (GMT).

Maar welk nut is een time stamp vraag/antwoord voor ons?

Als je een time stamp antwoord krijgt, weet je dat de host bereikbaar is, en aan de andere kant kun je met de Originate Stamp en de Receive Stamp bepalen wat de load (gebruik) van het netwerk is (her verschil is, natuurlijk, afhankelijk van de gebruikte kabels, kaarten, ...).

Ten slotte een tcpdump trace:

```
11:38:37.898253 Diablo > Diablo: icmp: time stamp query id 53763 seq 64548
(ttl 254, id 13170, len 40)
                4500 0028 3372 0000 fe01 8b60 7f00 0001
                7f00 0001 0d00 61fb d203 fc24 0211 c0ca
                0000 0000 0000 0000

11:38:37.898253 Diablo > Diablo: icmp: time stamp reply id 53763 seq 64548 :
org 0x211c0ca recv 0x211c0ca xmit 0x211c0ca (DF) (ttl 255, id 0, len 40)
                4500 0028 0000 4000 ff01 7dd2 7f00 0001
                7f00 0001 0e00 db43 d203 fc24 0211 c0ca
                0211 c0ca 0211 c0ca
```

ICMP Information Request / Reply (RFC 792):

```
"This message may
be sent with the source network in the IP header source and
destination address fields zero (which means "this" network).
The replying IP module should send the reply with the addresses
fully specified. This message is a way for a host to find out
the number of the network it is on. "



(NL: Dit bericht kan verstuurd worden met het bron netwerk in de
IP header bron en een doel adres van alleen nullen (wat "dit"
netwerk betekend). De antwoordende IP module zou het antwoord
moeten sturen met de adressen volledig gespecificeerd. Dit
bericht is een manier voor een host om te achterhalen op welk
netwerk deze zich bevindt.
```

Dus, een Information Request (Informatie Aanvraag, type 15) heeft als doel om het netwerk nummer van de verzendende host te achterhalen.

```
The address of the source in an information request message will be
the destination of the information reply message.  To form an
information reply message, the source and destination addresses
are simply reversed,..."
(NL: Het adres van de bron in een information request bericht zal het
doel zijn van de information reply. Om een antwoord te formuleren,
worden eenvoudig de bron en het doel omgewisseld...)
```

Met een Information Reply (informatie antwoord, type 16) neem je het bron IP van het Information Request als doel IP van het antwoord (simpel gezegd: je stuurt het antwoord naar de host die de vraag stelde). Als bron IP van het antwoord neem je het doel IP van de vraag...

Normaal gesproken is het zo dat de zender van een Information Request het doel adres op 0 zet (0 betekend "dit netwerk"). Maar de mogelijkheid bestaat ook om het doel en bron IP op 0 te zetten (bij het versturen van de vraag). In dit geval, krijgt de Information Reply het netwerk nummer van de host als bron en het doel adres veld, het bron adres in de request dus, is niet gelijk aan nul, dan zou het netwerk nummer van de host alleen teruggestuurd worden in het bron IP veld van het antwoord.

```
-------------------------------------------------------------
|         Type  |      Code      |   Checksum          |
-------------------------------------------------------------
|    Pointer     |       Sequence            |
-------------------------------------------------------------
```

Het lijkt er op dat je alleen een information request binnen het netwerk kunt sturen (zie hierboven), maar dit hoeft niet zo te zijn. Sommige besturingssystemen antwoorden ook op een information request waarbij het doel IP zich niet in hetzelfde netwerk bevindt. In een dergelijke information reply ontvangen we het IP van de host (en niet het netwerk nummer).

Ten slotte weer een korte tcpdump trace:

```
11:42:35.608253 Diablo > Diablo: icmp: information request (ttl 255,
 id 13170, len 28)
```

```
                          4500 001c 3372 0000 ff01 8a6c 7f00 0001
                          7f00 0001 0f00 1afc d603 0000
11:42:36.608253 Diablo > Diablo: icmp: information request (ttl 255,
 id 13170, len 28)
                          4500 001c 3372 0000 ff01 8a6c 7f00 0001
                          7f00 0001 0f00 19fc d603 0100
```

ICMP Address Mask Request / Reply (RFC 950):

```
    The Address Mask Request (type 17) has been described in another RFC,
    for more information look at RFC950 and not in RFC792. The meaning and
    use of an Address Mask Request is to get the subnetmask of a connected
    network.
    If a gateway receives such a request it should send back relevant
    information to the respective node (Address Mask Reply - type 18)



    (NL: Het Address Mask Request (type 17) is beschreven in een andere
    RFC, voor meer informatie, zie RFC792. Het doel en het gebruik van van
    een Adress Mask Request is om het subnetmasker van een verbonden
    netwerk te achterhalen. Als een gateway een dergelijk verzoek krijgt,
    zou deze relevante informatie terug moeten sturen (Address Mask Reply
    - type 18).
```

```
 ------------------------------------------------------------
 | Type  |      Code      |   Checksum                       |
 ------------------------------------------------------------
 | Flag         |       Sequence                             |
 ------------------------------------------------------------
 |               Address Mask                                 |
 ------------------------------------------------------------
```

Hiermee kun je niet alleen achterhalen welke hosts zich in het netwerk bevinden (welke online zijn) maar ook krijg je meer te weten over de netwerk configuratie met meer tests...

Tcpdump trace:

```
 11:45:26.678253 Diablo > Diablo: icmp: address mask request (ttl 254, id
  13170, len 32)
                 4500 0020 3372 0000 fe01 8b68 7f00 0001
                 7f00 0001 1100 edd7 dc03 2524 0000 0000
```

Ik hoop dat deze sectie je heeft laten zien dat er meerdere mogelijkheden zijn om informatie over een netwerk te krijgen, en dat dat niet alleen de "normale" pings hoeven te zijn... In de respectievelijke RFCs komen vaak dergelijke hints voor die je zou moeten overwegen als werkelijk de verschillende typen requests wil ondersteunen... Ontwikkelaars van ("echte") IDS’en zouden ook dergelijke onderwerpen moeten overwegen. Als ze ondersteund moeten worden, moet je ook overwegen dat het werkt (lees de RFCs). Maar zelfs dit is niet het einde, zoals je kunt lezen in de volgende sectie...

Nog meer informatie over het doel: Het gebruik van packet filters (of meer algemeen: firewalls) is zeker niet bijzonder tegenwoordig. Ook het gebruik van zogeheten firewall modules in IDS’en worden steeds vaker aangetroffen. Vaak heeft de aanvaller niet de mogelijkheden om sommige van de besproken scans te gebruiken, omdat ze worden geblokkeerd, gefilterd... Het doel van deze kleine sectie is om

mogelijkheden te laten zien waarmee je sommige van de filter/firewall regels van het doel kunt achterhalen.

Het principe van dit idee is erg simpel, daar je probeert ICMP fout berichten probeert uit te lokken, respectievelijk "illegale" pakketten verstuurd waarmee je conclusies kunt trekken over de set regels.

```
If the gateway or host processing a datagram finds a problem with
the header parameters such that it cannot complete processing the
datagram it must discard the datagram.  One potential source of
such a problem is with incorrect arguments in an option.  The
gateway or host may also notify the source host via the parameter
problem message.  This message is only sent if the error caused
the datagram to be discarded."

(NL: Als de gateway of de host een datagram verwerkt en een fout
ontdekt in de header parameters zodat deze het verwerken van het
datagram niet kan voltooien, dient deze het datagram te droppen.
Een potentiële bron van een dergelijk probleem is met incorrecte
argumenten in een optie. De gateway of host kan ook de bron host
op de hoogte stellen via een parameter probleem bericht. Dit bericht
wordt alleen verstuurd als de fout er voor zorgde dat het datagram
werd afgedankt.
```

Deze sectie komt uit RFC 792 en maakt deel uit van de beschrijving van het zogenoemde parameter problem (type 12). Zoals je kunt zien in deze sectie, kan een reden voor het bericht "Parameter Problem" een valse IP header zijn, we zouden dus een pakket met een valse IP header naar een host kunnen sturen en deze fout terug krijgen. Deze fout melding heeft verder het voordeel dat de ondersteuning van dit fout bericht "recommended" (aangeraden) is in RFC 1122:

```
A host SHOULD generate Parameter Problem messages.  An
incoming Parameter Problem message MUST be passed to the
transport layer, and it MAY be reported to the user.
(NL: Een host ZOU Parameter Problem berichten moeten
genereren.  Een inkomend Parameter Problem bericht MOET
doorgegeven aan de transport laag en MAG gerapporteerd
worden aan de gebruiker.)

DISCUSSION:
 The ICMP Parameter Problem message is sent to the
 source host for any problem not specifically covered by
 another ICMP message.  Receipt of a Parameter Problem
 message generally indicates some local or remote
 implementation error.   "


 (NL: Het ICMP Parameter Problem bericht wordt verstuurd
 naar de bron host voor ieder probleem waar geen ander
 ICMP bericht voor bestaat. De ontvanger van een Parameter
 Problem zal meestal een lokale of remote implementatie
 fout geven.)
```

Al met al, dit biedt erg goede mogelijkheden om hosts te detecteren in een netwerk (om genoemde redenen).

Aansluitend, met een referentie naar RFC 1812:

```
 4.3.3.5 Parameter Problem

A router MUST generate a Parameter Problem message for any error not
specifically covered by another ICMP message.  The IP header field or
IP option including the byte indicated by the pointer field MUST be
included unchanged in the IP header returned with this ICMP message.
Section [4.3.2] defines an exception to this requirement.  "

NL: Een router MOET een Parameter Parameter Problem bericht genereren,
voor iedere fout die niet specifiek wordt gedekt door andere ICMP
berichten. Het IP header veld of IP optie, inclusief de byte die door het
pointer veld wordt aangegeven MOET onveranderd in het IP header veld
worden geretourneerd met dit ICMP bericht. Sectie [4.3.2] definiëert
een uitzondering op deze verplichting.
```

Niettemin, verschillende routers interpreteren deze sectie (en ook andere) verschillend, waardoor het niet duidelijk is dat een Parameter Problem wordt gegenereerd.

Een IDS (resp. firewalls) zouden velden in de IP header toch moeten controleren, daar het kan voorkomen dat je een pakket ontvangt met een valse header, maar dit zou zelden moeten voorkomen. Een aanvaller die een heel netwerk scant met deze methode (of een specifieke IP range) weet dat de firewall/packet filter dit fout bericht... niet filterd. Maar als je geen Parameter Problem bericht krijgt, weet je in ieder geval dat dit bericht wordt geblokkeerd.

Deze methode om de set regels te achterhalen is slechts de eerste stap (de methode vooral een alternatief voor een "normale" ping). Om een zo exact mogelijke access control list (ACL) van mogelijke filtering/firewall te krijgen, hebben we meer informatie nodig over de topologie. Hiervoor zou je, bijvoorbeeld, de verschillende ICMP bericht typen naar losse hosts moeten sturen (met valse IP header) en dan wachten op de notificatie van een Parameter Problem. Als er een notificatie komt van een Parameter Problem op host "X", dan betekend dit voor ons dat host "X" dit ICMP bericht type niet filtert (en dat de respectievelijke host bereikbaar is). Aansluitend betekend dit voor ons dat de foutieve informatie in de IP header niet wordt gecontroleerd. Als we geen Parameter Problem krijgen, kunnen we verschillende conclusies trekken. Aan de ene kant, is het mogelijk dat een filter het ICMP bericht type filtert/blokkeert en aan de andere kant dat het mogelijk is dat de router de header controleert en dus deze anomalie niet laat passeren, etc... Zoals je kunt zien, met beide resultaten kun je conclusies trekken over mogelijke filter regels, en bovendien krijg je een brede representatie van wat er wordt gebruikt en wat niet. Verdere mogelijkheden zouden zijn om de gebruikte protocollen te variëren (TCP, UDP, ...) zodat je verdere regels kunt achterhalen. Zo is er de mogelijkheid dat bijvoorbeeld een bepaald protocol wordt geblokkeerd/gefilterd.

Ik denk dat het idee van het Parameter Problem nu duidelijk is zodat we kunnen gaan kijken naar verdere fout berichten.

De volgende sectie komt weer uit RFC 1812 en beschrijft wat de Destination Unreachable error is en wat de reden van deze fout kan zijn:

```
The ICMP Destination Unreachable message is sent by a router in
response to a packet which it cannot forward because the destination
(or next hop) is unreachable or a service is unavailable.  Examples
of such cases include a message addressed to a host which is not
there and therefore does not respond to ARP requests, and messages
addressed to network prefixes for which the router has no valid
route.
```

(NL: Het ICMP Destination Unreachable bericht wordt verstuurd door een router in antwoord op een pakket dat hij niet kan doorsturen omdat de bestemming (of volgende hop) niet bereikbaar is, of een service niet beschikbaar. Voorbeelden van zulke gevallen bevatten een bericht naar een host welke er niet is en daardoor niet reageerd op ARP vragen, en berichten die zijn geadresseerd aan netwerk prefixes waarvoor de router niet over een geldige route beschikt.)

A router MUST be able to generate ICMP Destination Unreachable messages and SHOULD choose a response code that most closely matches the reason the message is being generated.

(NL: Een router MOET in staat zijn om ICMP Destination Unreachable berichten te genereren en ZOU een antwoord code moeten kiezen die het dichtst bij de reden aansluit waarom het bericht wordt gegenereerd.)

0 = Network Unreachable - generated by a router if a forwarding path (route) to the destination network is not available;

(NL: Netwerk onbereikbaar - wordt gegenereerd door een router als opgegeven pad (route) naar het doel netwerk niet beschikbaar is)

1 = Host Unreachable - generated by a router if a forwarding path (route) to the destination host on a directly connected network is not available (does not respond to ARP);

(NL: Host niet bereikbaar - gegenereerd door een router als het opgegeven pad naar de doelhost zich niet op een direct verbonden netwerk bevindt (reageert niet op ARP)

2 = Protocol Unreachable - generated if the transport protocol designated in a datagram is not supported in the transport layer of the final destination;

(NL: Protocol niet bereikbaar - gegenereerd als het transport protocol aangeduid in het datagram niet ondersteund wordt door de transportlaag van de uiteindelijke bestemming)

3 = Port Unreachable - generated if the designated transport protocol (e.g., UDP) is unable to demultiplex the datagram in the transport layer of the final destination but has no protocol mechanism to inform the sender;

(NL: Poort Onbereikbaar - gegenereerd als het gebruikte transport protocol (bijv. UDP) niet in staat is om het datagram in de transport laag van het doel uit te pakken maar niet over een mechanisme beschikt om de zender te informeren.)

4 = Fragmentation Needed and DF Set - generated if a router needs to fragment a datagram but cannot since the DF flag is set;

(NL:Fragmentatie Nodig en DF Set - gegenereerd als een router een datagram moet fragmenteren maar dit niet kan omdat de DF flag gezet is.)

5 = Source Route Failed - generated if a router cannot forward a packet to the next hop in a source route option;

```
     (NL:Bron Route Faalde - gegenereerd als een router een pakket
      niet kan doorsturen naar de volgende hop in een bron route optie.)

 6 = Destination Network Unknown - This code SHOULD NOT be generated
      since it would imply on the part of the router that the
      destination network does not exist (net unreachable code 0
      SHOULD be used in place of code 6);

     (NL:Doel Netwerk Onbekend - Deze code ZOU NIET gegenereerd worden
      daar het impliceert dat, voor wat de router betreft, het doel-
      netwerk niet bestaat. (net unreachable code 0 ZOU gebruikt moeten
      worden in plaats van code 6.)

 7 = Destination Host Unknown - generated only when a router can
      determine (from link layer advice) that the destination host
      does not exist;

     (NL: Doel Host Onbekend - alleen gegenereerd als een router kan
      bepalen (van link laag advies) dat de doel host niet bestaat.)

 11 = Network Unreachable For Type Of Service - generated by a router
       if a forwarding path (route) to the destination network with the
       requested or default TOS is not available;

      (NL: Netwerk Onbereikbaar Voor Type Of Service - gegenereerd door
       een router als een doorstuur pad (pad) naar het doel netwerk met
       de gevraagde of default TOS niet beschikbaar is.)

 12 = Host Unreachable For Type Of Service - generated if a router
       cannot forward a packet because its route(s) to the destination
       do not match either the TOS requested in the datagram or the
       default TOS (0).         "

      (NL: Host onbereibaar voor Type Of Service - gegenereerd als een
       router een pakket niet kan doorsturen omdat zijn route(s) naar
       de doel host niet overeenkomen met ofwel de TOS gevraagd in het
       datagram of de default TOS (0).)
```

Als we nu, bijvoorbeeld, een pakket proberen te sturen naar een poort welke een protocol gebruikt dat niet bestaat, zou er in feite een notificatie moeten komen over een Destination Unreachable met code waarde 2 (Protocol Unreachable). Verder, met dit voorbeeld zou je eerst moeten weten welke protocollen er worden "toegestaan". Dit kun je uitvinden door te kijken in /etc/protocols. Na de installatie van een van mijn hosts zag de /etc/protocols er als volgt uit:

```
 -------------- /etc/protocols ----------------
# /etc/protocols:
# $Id: protocols,v 1.2 2001/01/29 17:29:30 notting Exp $
#
# Internet (IP) protocols
#
#       from: @(#)protocols     5.1 (Berkeley) 4/17/89
#
# Updated for NetBSD based on RFC 1340, Assigned Numbers (July 1992).
#
# See also http://www.isi.edu/in-notes/iana/assignments/protocol-numbers

ip      0       IP              # internet protocol, pseudo protocol number
#hopopt 0       HOPOPT          # hop-by-hop options for ipv6
icmp    1       ICMP            # internet control message protocol
igmp    2       IGMP            # internet group management protocol
```

```
ggp     3       GGP             # gateway-gateway protocol
ipencap 4       IP-ENCAP        # IP encapsulated in IP (officially ‘‘IP’’)
st      5       ST              # ST datagram mode
tcp     6       TCP             # transmission control protocol
cbt     7       CBT             # CBT, Tony Ballardie
egp     8       EGP             # exterior gateway protocol
igp     9       IGP             # any private interior gateway
                                # (Cisco: for IGRP)
bbn-rcc 10      BBN-RCC-MON     # BBN RCC Monitoring
nvp     11      NVP-II          # Network Voice Protocol
pup     12      PUP             # PARC universal packet protocol
argus   13      ARGUS           # ARGUS
emcon   14      EMCON           # EMCON
xnet    15      XNET            # Cross Net Debugger
chaos   16      CHAOS           # Chaos
udp     17      UDP             # user datagram protocol
mux     18      MUX             # Multiplexing protocol
dcn     19      DCN-MEAS        # DCN Measurement Subsystems
hmp     20      HMP             # host monitoring protocol
prm     21      PRM             # packet radio measurement protocol
xns-idp 22      XNS-IDP         # Xerox NS IDP
trunk-1 23      TRUNK-1         # Trunk-1
trunk-2 24      TRUNK-2         # Trunk-2
leaf-1  25      LEAF-1          # Leaf-1
leaf-2  26      LEAF-2          # Leaf-2
rdp     27      RDP             # "reliable datagram" protocol
irtp    28      IRTP            # Internet Reliable Transaction Protocol
iso-tp4 29      ISO-TP4         # ISO Transport Protocol Class 4
netblt  30      NETBLT          # Bulk Data Transfer Protocol
mfe-nsp 31      MFE-NSP         # MFE Network Services Protocol
merit-inp       32    MERIT-INP       # MERIT Internodal Protocol
sep     33      SEP             # Sequential Exchange Protocol
3pc     34      3PC             # Third Party Connect Protocol
idpr    35      IDPR            # Inter-Domain Policy Routing Protocol
xtp     36      XTP             # Xpress Tranfer Protocol
ddp     37      DDP             # Datagram Delivery Protocol
idpr-cmtp       38    IDPR-CMTP       # IDPR Control Message Transport Proto
tp++    39      TP++            # TP++ Transport Protocol
il      40      IL              # IL Transport Protocol
ipv6    41      IPv6            # IPv6
sdrp    42      SDRP            # Source Demand Routing Protocol
ipv6-route      43    IPv6-Route      # Routing Header for IPv6
ipv6-frag       44    IPv6-Frag       # Fragment Header for IPv6
idrp    45      IDRP            # Inter-Domain Routing Protocol
rsvp    46      RSVP            # Resource ReSerVation Protocol
gre     47      GRE             # Generic Routing Encapsulation
mhrp    48      MHRP            # Mobile Host Routing Protocol
bna     49      BNA             # BNA
ipv6-crypt      50    IPv6-Crypt      # Encryption Header for IPv6
ipv6-auth       51    IPv6-Auth       # Authentication Header for IPv6
i-nlsp  52      I-NLSP          # Integrated Net Layer Security TUBA
swipe   53      SWIPE           # IP with Encryption
narp    54      NARP            # NBMA Address Resolution Protocol
mobile  55      MOBILE          # IP Mobility
tlsp    56      TLSP            # Transport Layer Security Protocol
skip    57      SKIP            # SKIP
ipv6-icmp       58    IPv6-ICMP       # ICMP for IPv6
ipv6-nonxt      59    IPv6-NoNxt      # No Next Header for IPv6
ipv6-opts       60    IPv6-Opts       # Destination Options for IPv6
#       61                      # any host internal protocol
cftp    62      CFTP            # CFTP
```

```
#       63                      # any local network
sat-expak       64      SAT-EXPAK       # SATNET and Backroom EXPAK
kryptolan       65      KRYPTOLAN       # Kryptolan
rvd     66      RVD             # MIT Remote Virtual Disk Protocol
ippc    67      IPPC            # Internet Pluribus Packet Core
#       68                      # any distributed file system
sat-mon 69      SAT-MON         # SATNET Monitoring
visa    70      VISA            # VISA Protocol
ipcv    71      IPCV            # Internet Packet Core Utility
cpnx    72      CPNX            # Computer Protocol Network Executive
cphb    73      CPHB            # Computer Protocol Heart Beat
wsn     74      WSN             # Wang Span Network
pvp     75      PVP             # Packet Video Protocol
br-sat-mon      76      BR-SAT-MON      # Backroom SATNET Monitoring
sun-nd  77      SUN-ND          # SUN ND PROTOCOL-Temporary
wb-mon  78      WB-MON          # WIDEBAND Monitoring
wb-expak        79      WB-EXPAK        # WIDEBAND EXPAK
iso-ip  80      ISO-IP          # ISO Internet Protocol
vmtp    81      VMTP            # Versatile Message Transport
secure-vmtp     82      SECURE-VMTP     # SECURE-VMTP
vines   83      VINES           # VINES
ttp     84      TTP             # TTP
nsfnet-igp      85      NSFNET-IGP      # NSFNET-IGP
dgp     86      DGP             # Dissimilar Gateway Protocol
tcf     87      TCF             # TCF
eigrp   88      EIGRP           # Enhanced Interi
```

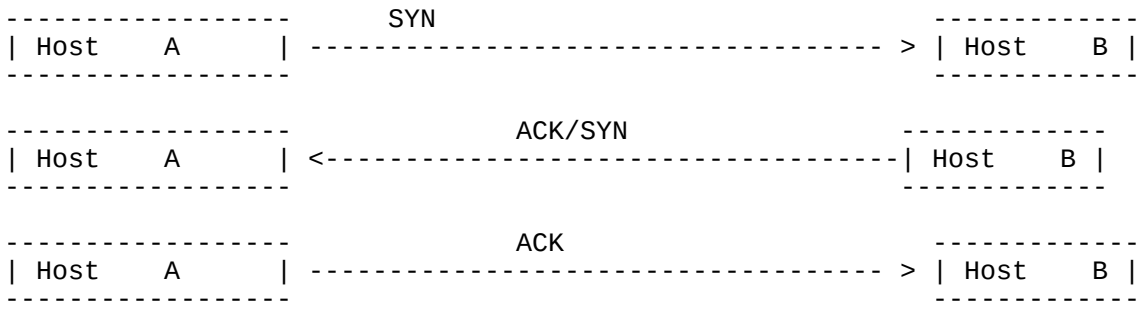
Wat als de host geen Protocol Unreachable terugstuurt (al gebruikte je een protocol dat niet bestaat)? Dit kan twee redenen hebben. Ten eerste zou het kunnen dat je een AIX, HP-UX of Digital Unix machine bent tegengekomen of als de regelset op de host je niet toestaat om deze poorten te benaderen. Dus, ten eerste, controleer welk soort host je scant (oa mogelijk met een fingerprint OS detectie) of anders kun je aannemen dat het gefilterd/geblockt wordt.

Opmerking: De detectie van een dergelijke aanval is erg simpel (en hoort tot de sectie van Protocol Anomaly Detectie). Anomaly Detectie zal worden besproken in het volgende hoofdstuk (mogelijkheden voor analyse), nu is het voldoende om te weten dat verkeer wordt doorzocht op "afwijkingen" en het gebruik van een niet-bestaand protocol behoort tot deze "afwijkingen".

- Denial of Service (DoS)
DoS (Denial of Service) zijn er meestal op gericht om de doel server te bevriezen zodat deze een tijd niet bereikbaar is. Er zijn erg veel technieken waarmee je de bronnen van een doel host kunt "uitputten". In het volgende stel ik de meest voorkomende voor:

ICMP Flooding:
ICMP Flooding "gebruikt" ICMP. Als een host een ICMP echo request krijgt, zal deze normaal gesproken om te antwoorden met een ICMP echo reply. Dit gedrag wordt gebruikt met ICMP Flooding: Als je het slachtoffer teveel echo requests stuurt, zullen zijn bronnen aan het werk gaan om deze requests te beantwoorden (met echo replies). Daar het IP van de aanvaller meestal gespoofed is, krijgt hij niet de antwoorden, maar iemand anders.

SYN Flooding:
Om SYN Flooding te begrijpen, presenteer ik hier nogmaals de "normale" three-way-handshake:



Host A stuurt Host B een SYN om aan te geven dat hij een connectie wil, B antwoord met een ACK/SYN en wacht op de laatste ACK waarmee de connectie compleet zou zijn. Maar wat als de laatste ACK niet verstuurd wordt? Als B zijn SYN/ACK terugstuurt wacht het (zoals genoemd) op de ACK van A, tot die tijd zal het worden geplaatst in de connectie wachtrij (connection queue) van B. Als de connectie compleet is (A stuurt B een ACK), zal deze uit de wachtrij verwijderd worden. Maar daar het IP adres gespoofed is, krijgt B nooit een ACK (zo zou het moeten zijn). Zo kun je dus de connectie wachtrij "vullen" tot de host geen verdere connecties meer kan maken.

UDP Flooding:
Met deze flooding aanval wordt de doel server overspoelt met UDP pakketten. Als je een UDP pakket naar een poort van de doel server stuurt, zal deze eerst controleren welke service verantwoordelijk is voor deze "vraag". Je kiest vervolgens random poorten naar welke je de pakketten stuurt om de waarschijnlijkheid van een "Port Unreachable" te verhogen. Als resultaat van een dergelijke flood lijdt de performance van het netwerk er (meestal) behoorlijk onder.

Land:
Later zul je een filter zien dat detecteerd of er een Land attack is en tegenmaatregelen kan treffen, maar eerst de eigenlijke aanval. Een Land aanval heeft gelijke bron en doel IPs. Wanneer bijvoorbeeld SYN pakketten worden verstuurd (met hetzelfde bron/doel IP) naar een open poort zal er een race conditie optreden op de host van het slachtoffer welke kan leiden tot het bevriezen van het hele syteem. Hier is een trace van een Land attack:

```
23/06/02  23:12:48  194.157.1.1  80  -> 194.157.1.1
23/06/02  23:14:57  194.157.1.1  31337  ->     194.157.1.1
```

Zoals je kunt zien, zijn de bron en doel IPs identiek.

```
12:35:26.916369 192.168.38.110.135>192.168.38.110.135: udp 46[tos0x3,ECT,CE]
        4503 004a 96ac 0000 4011 15c7 c0a8 266e
       c0a8 266e 0087 0087 0036 8433 6920 616d
       206c 616d 6520 646f 7320 6b69 6420 6275
       7420 6920 7265
12:35:26.916566 192.168.38.110.135>192.168.38.110.135: udp 46[tos0x3,ECT,CE]
       4503 004a 2923 0000 4011 8350 c0a8 266e
       c0a8 266e 0087 0087 0036 8433 6920 616d
       206c 616d 6520 646f 7320 6b69 6420 6275
       7420 6920 7265
12:35:26.916682 192.168.38.110.135>192.168.38.110.135:udp 46[tos0x3,ECT,CE]
       4503 004a 50a0 0000 4011 5bd3 c0a8 266e
       c0a8 266e 0087 0087 0036 8433 6920 616d
       206c 616d 6520 646f 7320 6b69 6420 6275
       7420 6920 7265
```

De aanval die je hier ziet wordt ook wel Snork genoemd.

Teardrop:
Hier wordt de mogelijkheid van fragmentatie van IP pakketten gebruikt. Als je kijkt in de beschrijving van de scans, is er een fragmentatie als de pakketten groter zijn dan de grootte limiet, deze grootte limitatie wordt ook wel MTU (Maximum Transmission Unit) genoemd. Met deze aanval overlappen de fragmenten elkaar en als een resultaat hiervan, kregen veel OSsen problemen en vaak crashte het systeem.

```
10:13:32.104203 10.10.10.10.53>192.168.1.3.53: udp 28(frag 242:36@0+)(ttl64)
       4500 0038 00f2 2000 4011 8404 0a0a 0a0a
       c0a8 0103 0035 0035 0024 0000 0000 0000
       0000 0000 0000 0000 0000 0000 0000 0000
        0000 0000 0000

10:13:32.104272 10.10.10.10 >192.168.1.3: udp 28(frag 242:4@24)(ttl 64)
       4500 0018 00f2 0003 4011 a421 0a0a 0a0a
       c0a8 0103 0035 0035 0024 0000 0000 0000
       0000 0000 0000 0000 0000 0000 0000
```

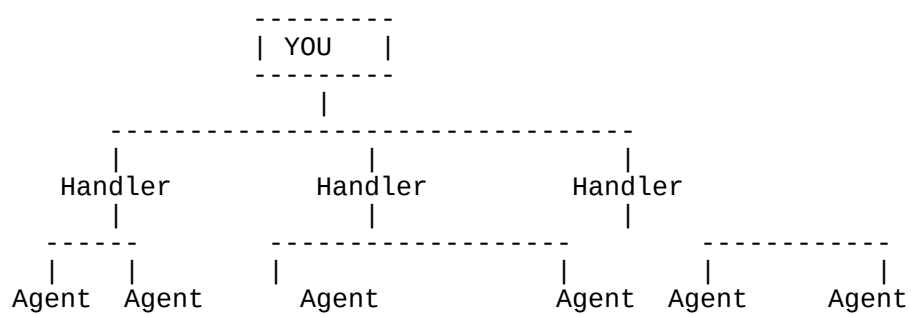
Ping of Death:
Ook hier speelt de fragmentatie van de IP pakketten een rol. Hier zal ik in (iets) meer detail op ingaan op fragmentatie. Zoals eerder gezegd, betekend fragmentatie dat de grootte van datagrammen "gereduceerd" is, waarbij de fragmenten niet groter zijn dan de MTU. Bij fragmenteren moet je je bedenken dat je niet willekeurig kunt fragmenteren, respectievelijk dat bepaalde velden in alle velden moeten voorkomen. Dus, ieder fragment dient bijvoorbeeld de IP protocol header bevatten om de juiste route te kiezen. opdat de router de fragmenten kan herbouwen tot een datagram ontvangt ieder fragment een 16 bits flag (van het originele, "grote" datagram). Met de 16 bit flag is het later mogelijk om de fragmenten te sorteren tot het juiste datagram. Aansluitend is er een fragment offset welke aangeeft op welke positie het fragment zich bevondt in het originele datagram. Maar de positie is in 8 octet units (de positie wordt gemeten in 8 octet units). Aansluitend, het "more-bit" geeft aan of er verdere fragmenten van het datagram volgt of niet. Als deze is gezet op 1 zullen er meer volgen, als deze staat op 0, was dit het laatste fragment van het datagram. Nu komen we bij de eigenlijke ping-of-death aanval.
Ping-of-death krijgt een offset van het laatste fragment welke is: offset + fragment grootte > 65535 bytes. Dus is het mogelijk om de interne 16 bit variabelen te overvloeden, wat vervolgens resulteerd in, bijvoorbeeld, een systeem crash.

```
17:43:58.431 pinger > target: icmp echo request (frag 4321: 380@0+)
17:43:58.431 pinger > target: (frag 4321: 380@2656+)
17:43:58.431 pinger > target: (frag 4321: 380@3040+)
17:43:58.431 pinger > target: (frag 4321: 380@3416+)
```

Smurf:
Een ping wordt gestuurd naar het broadcast adres,of, beter gezegd worden er vele pings gestuurt. Pakketten verzonden naar het broadcast adres worden verstuurd naar alle hosts in het netwerk. Als je veel pings (ICMP echo requests) stuurt naar het broadcast adres (bijvoorbeeld 10000 per seconde) en je hebt een relatief groot netwerk met 1000 hosts, betekend dit dat er 10000 * 1000 ICMP echo replies per seconde naar de host van het slachtoffer worden gestuurd, dat betekend dus 10000000 ICMP echo replies.

```
09:28:28.666073 179.135.168.43>256.256.30.255: icmp: echo request (DF)
       4500 001c c014 4000 1e01 6172 b387 a82b
       c0a8 1eff 0800 f7ff 0000 0000 0000 0000
       0000 0000 0000 0000 0000 0000 0000
09:28:28.696073 68.90.226.250>256.256.30.255: icmp: echo request (DF)
       4500 001c c015 4000 1e01 95cf 445a e2fa
       c0a8 1eff 0800 f7ff 0000 0000 3136 3803
       3133 3503 3137 3907 696e 2d61 6464
09:28:28.726073 138.98.10.247>256.256.30.255:  icmp: echo request (DF)
       4500 001c c016 4000 1e01 27ca 8a62 0af7
       c0a8 1eff 0800 f7ff 0000 0000 0332 3236
       3938 0331 3638 0769 6e2d 6164 6472
09:28:28.756073 130.113.202.100 > 256.256.30.255: icmp: echo request (DF)
       4500 001c c017 4000 1e01 704c 8271ca64
       c0a8 1eff 0800 f7ff 0000 0000 0231 3002
       3938 0331 3338 0769 6e2d 6164 6472
       ...
```

Sinds enige tijd bestaan er ook DDoS aanvallen (Distributed Denial of Service). Zoals de naam suggereert is dit een gedistribueerde/genetwerkte DoS aanval. De aanvaller (client) zoekt naar andere hosts/netwerken... welke eenvoudig te exploiteren zijn. Deze eerst geïnfecteerde hosts worden de handlers genoemd. Handlers infecteren verdere hosts/netwerken, deze geïnfecteerde hosts worden dan agents genoemd. Als datagram:



Later voeren de agenten aanvallen uit.

Dit is het eerste artikel van 2. We zullen verder gaan in het volgende nummer van LinuxFocus.

2005-01-14, generated by lfparser_pdf version 2.51